Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Курсовая работа по дисциплине

«Базы данных»

Вариант 38 (2)

Тема: «Разработка базы данных экспорта товаров»

Выполнил: студент группы РИС-22-1Б

Головин М.А.

Проверил: доцент кафедры ИТАС

Петренко А.А.

г. Пермь – 2024

РЕФЕРАТ

Объектом исследования является система работы с экспортом товаров и проверки товаров в таможнях.

Цель работы – разработать реляционную базу данных и приложение к ней для системы отслеживания проверки грузов на таможнях.

В процессе работы проводилось моделирование и разработка структуры реляционной базы данных.

В результате работы была создана реляционная база данных и приложение для взаимодействия с ней.

ВВЕДЕНИЕ

В современном мире глобализация и международная торговля играют ключевую роль в экономическом развитии стран и предприятий. Успешное ведение экспортных операций требует точного и систематизированного подхода к учету и анализу данных о товарах и их движении. Одним из эффективных инструментов для этого является создание базы данных экспорта товаров.

Целью работы является исследование принципов и этапов создания базы данных экспорта товаров, а также анализ ее значимости для оптимизации экспортных процессов. В работе будут рассмотрены основные методы и технологии, используемые для разработки таких баз данных, их структура и функциональные возможности.

Актуальность темы обусловлена растущей потребностью в эффективном управлении экспортной деятельностью, что требует высококачественной информации и аналитических инструментов для принятия обоснованных решений. Создание базы данных экспорта товаров позволяет повысить оперативность и точность управления.

Задачи:

1. Анализ предметной области;
2. Проектирование схемы базы данных;
3. Разработка приложения для просмотра и управления данными в базе;
4. Тестирование программы с базой данных на соответствие требованиям;

**1 Анализ предметной области**

Требуется разработать реляционную базу данных и приложение для управления информацией экспорта товаров и расчёта времени проверки поставок. Основные критерии выбора СУБД - актуальность, поддержка сообщества, наличие русскоязычной документации, функциональность и удобство в использовании.

Предметная область включает сущности: "Страна", "Товар", "Тип товара", "Поставка", "Таможня". Каждая таможня имеет уникальное название. У поставок есть содержимое, состоящее из товаров определённого типа, скорость проверки поставок зависит от товаров и конкретной таможни.

Сущности:

1. Товар
2. Тип товара
3. Страна
4. Поставка
5. Таможня
6. График
7. Процесс

Не может быть поставки без товаров, к каждой таможне привязан процесс, определяющий скорость проверки определённого типа товара.

Приложение позволяет расчитать количество часов, которое потребуется для проверки определённой поставки, указанной полльзователем.

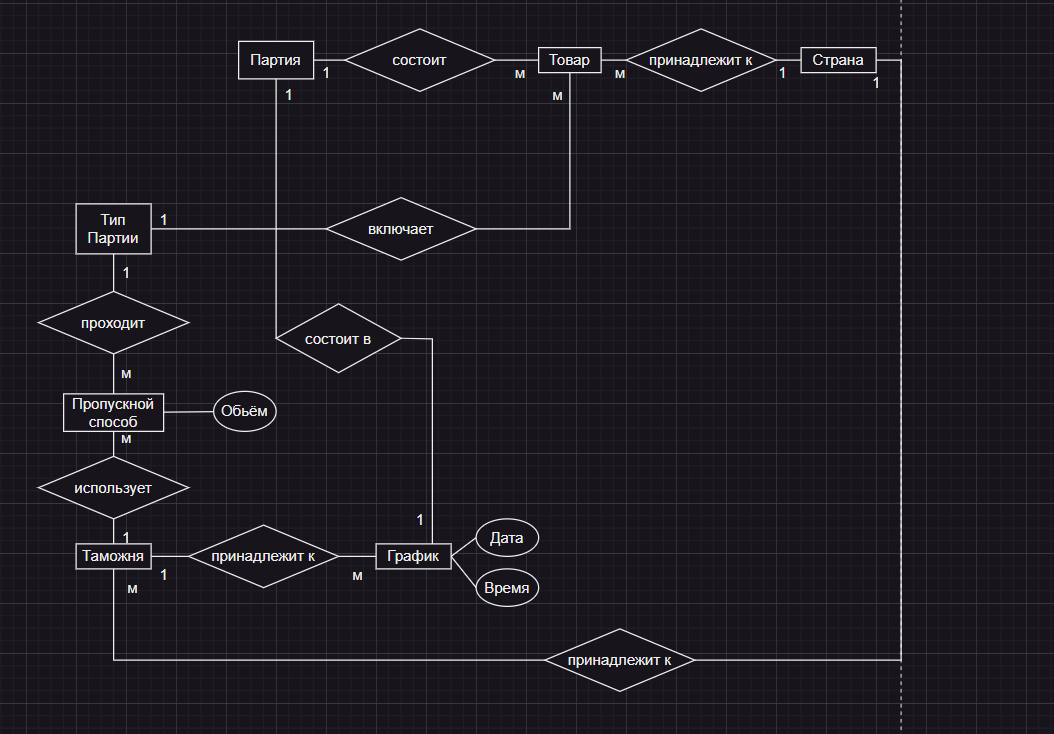
**Вывод**

В результате анализа предметной области были выявлены 7 сущностей, связи между ними и требования к приложению.

**Проектирование схемы базы данных**

Концептуальная модель в нотации Питера Чена:

База данных состоит из 7 таблиц: «Таможня», «Поставка», «Тип товара», «Товар», «Скорость проверки», «Расписание», «Страна».



*Рисунок 1 – Концептуальная модель базы данных «Экспорт товаров».*

Каждая партия включает в себя несколько товаров, которые в свою очередь включают в себя тип товара. К каждой таможне относятся несколько измерений скорости проверки определённого типа товара и график поступления партий.

**Выбор реляционной СУБД**

Выбор РСУБД в 2024 году является непростой задачей – существует огромное множество похожих систем управления с разным набором преимуществ. Автор решил провести сравнительный анализ среди нескольких популярных РСУБД: *SQL Server*, *MySQL* и *Microsoft Access*, приведенный в таблице 1.

Таблица 1 – сравнительная характеристика СУБД

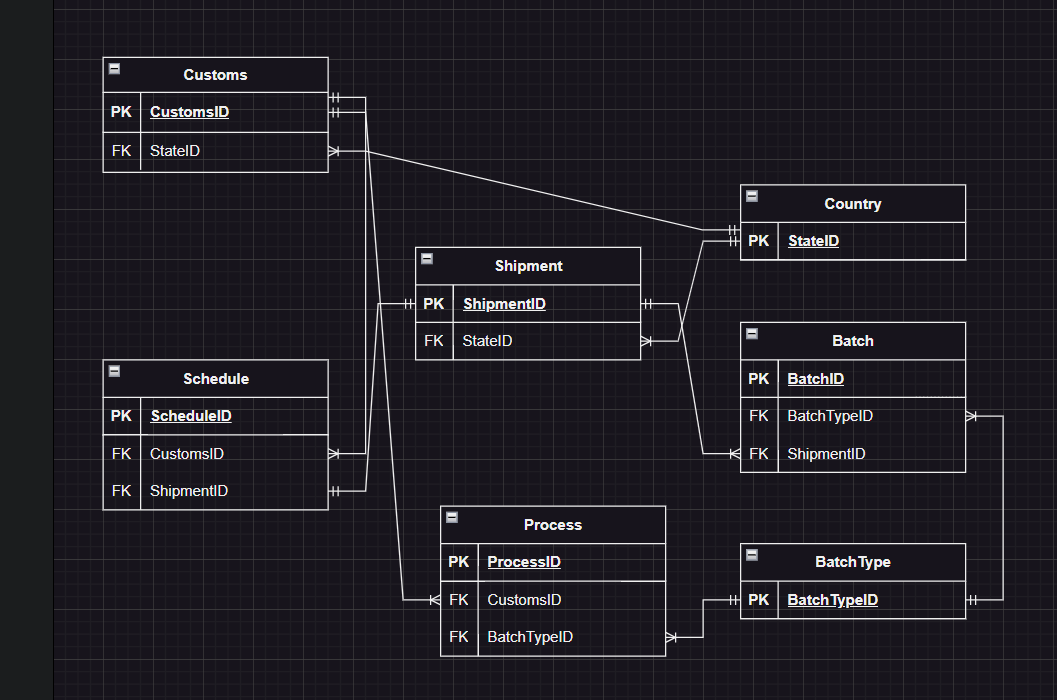
|  |  |  |  |
| --- | --- | --- | --- |
| Критерий сравнения | Реляционные системы управления базами данных | | |
| *SQL Server* | *MySQL* | *Microsoft Access* |
| Производительность | + | + | - |
| Расширяемость | + | + | - |
| Сложность настройки | - | + | + |
| Поддержка стандартов | + | + | - |
| Оптимизация запросов | + | + | - |
| Интерфейс | + | + | - |
| Цена | - | + | +- |

Из таблицы сравнения видно, что *MySQL* выигрывает по всем параметрам у своих конкурентов.

Таким образом, в качестве РСУБД автор выбрал *MySQL*, опережающий соперников по большему числу критериев*.*

Логическая модель в нотации Мартина:

Логическая модель состоит из отношений со значимыми атрибутами, связями с другими отношениями и ограничений. Каждая сущность содержит первичный ключ «*ID*», уникально идентифицирующий кортеж.



*Рисунок 2 – Логическая модель базы данных «Экспорт товаров».*

«*Customs*»:

* Внешний ключ «*StateID*» к сущности «*State*»

«*Schedule*»:

* Внешний ключ «*CustomsID*» к сущности «*Customs*»
* Внешний ключ «*ShipmentID*» к сущности «*Shipment*»

«*Process*»:

* Внешний ключ «*CustomsID*» к сущности «*Customs*»
* Внешний ключ «*BatchTypeID*» к сущности «*BatchType*»

«*Shipment*»:

* Внешний ключ «*StateID*» к сущности «*State*»

«*Batch*»:

* Внешний ключ «*BatchTypeID*» к сущности «*BatchType*»
* Внешний ключ «*ShipmentID*» к сущности «*Shipment*»

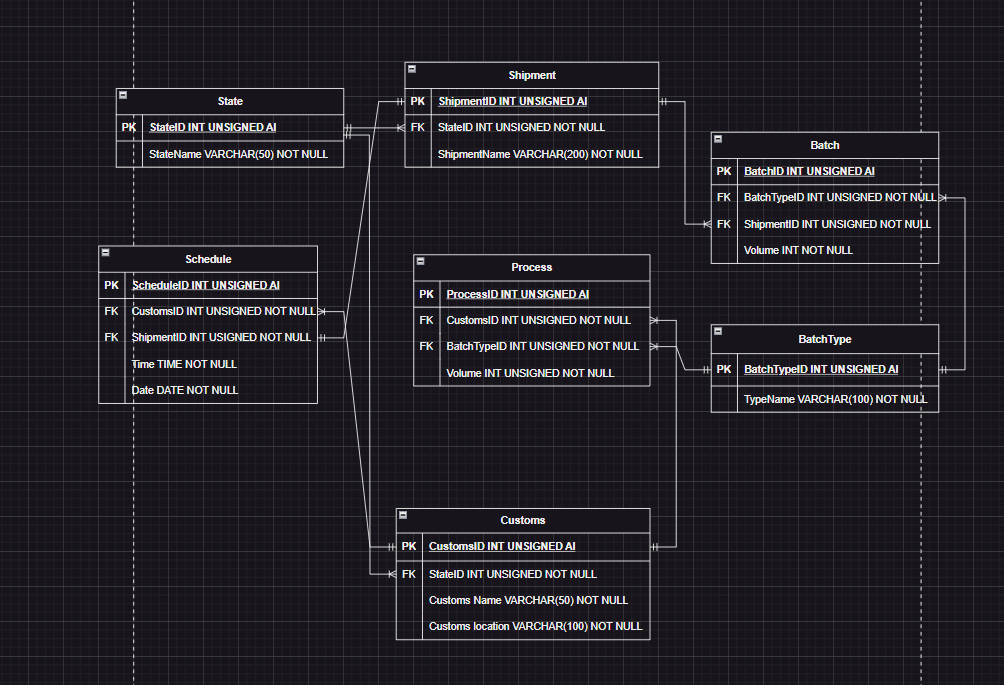
На рисунке 2 продемонстрирована построенная модель в нотации Мартина.

Таким образом, разработана логическая модель базы данных в виде диаграммы в нотации Мартина по выявленным сущностям и связям, удовлетворяющая требованиям 3 нормальной формы.

Физическая модель в нотации Мартина:

Физическая модель дополняет логическую, а также содержит типы атрибутов, незначимые атрибуты и функции, триггеры, представления.

Первичные ключи имеют тип *INT* с атрибутом *UNSIGNED*, ненулевое целочисленное поле типа “первичный ключ” с автоматическим инкрементом значения, внешние ключи – тип *INT*.



*Рисунок 3 – Физическая модель базы данных «Экспорт товаров».*

Таким образом, разработана физическая модель базы данных в виде диаграммы в нотации Мартина по выявленным сущностям и связям.

**Вывод**

При моделировании были составлены концептуальная, логическая и физическая модели базы данных, проверено выполнение требований третьей нормальной формы спроектированной системой.

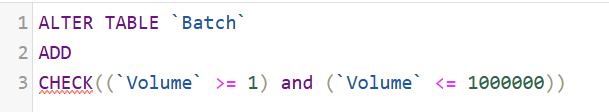
В результате сравнительного анализа популярных СУБД выбрана *MySQL*.

**Разработка приложения и базы данных «Экспорт товаров»**

Разработка базы данных:

Для реализации базы данных был выбран инструмент для проектирования баз данных «*phpMyAdmin*».

В таблице «Batch», в сущности установлено ограничение «*CHECK*», которое не даёт выставить значение столбца «*Volume*» меньше единицы и больше миллиона. Добавление ограничения было сделано по *SQL* команде.



*Рисунок 4 – SQL команда для добавления ограничения.*

Разработка приложения:

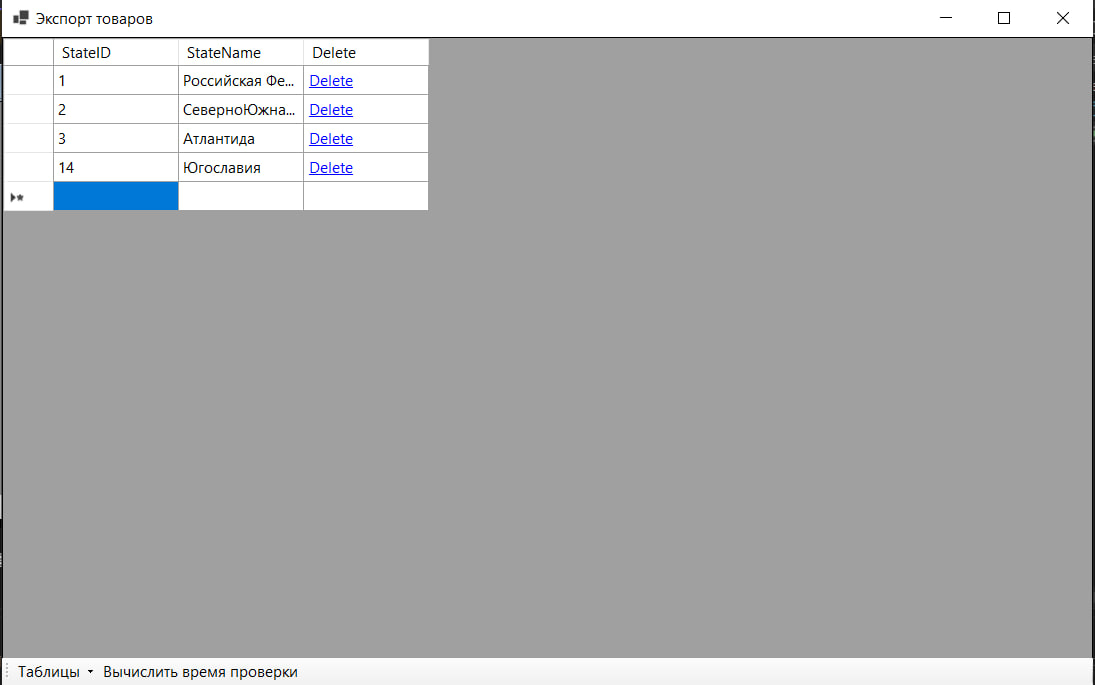
Приложение разработано с использованием *WindowsForms* на языке *C#*.

В качестве фреймворка выбран *WindowsForms* по ряду причин:

1. Стандартизированная структура (разделение на множество атомарных файлов)
2. Готовая библиотека для создания графического интерфейса
3. Мощная поддержка сообщества
4. Подробная документация и множество учебных материалов
5. Автор имеет опыт применения технологии

Приложение создается по методологии объектно-ориентированного программирования.

На рисунке 5 изображён интерфейс программы при её запуске

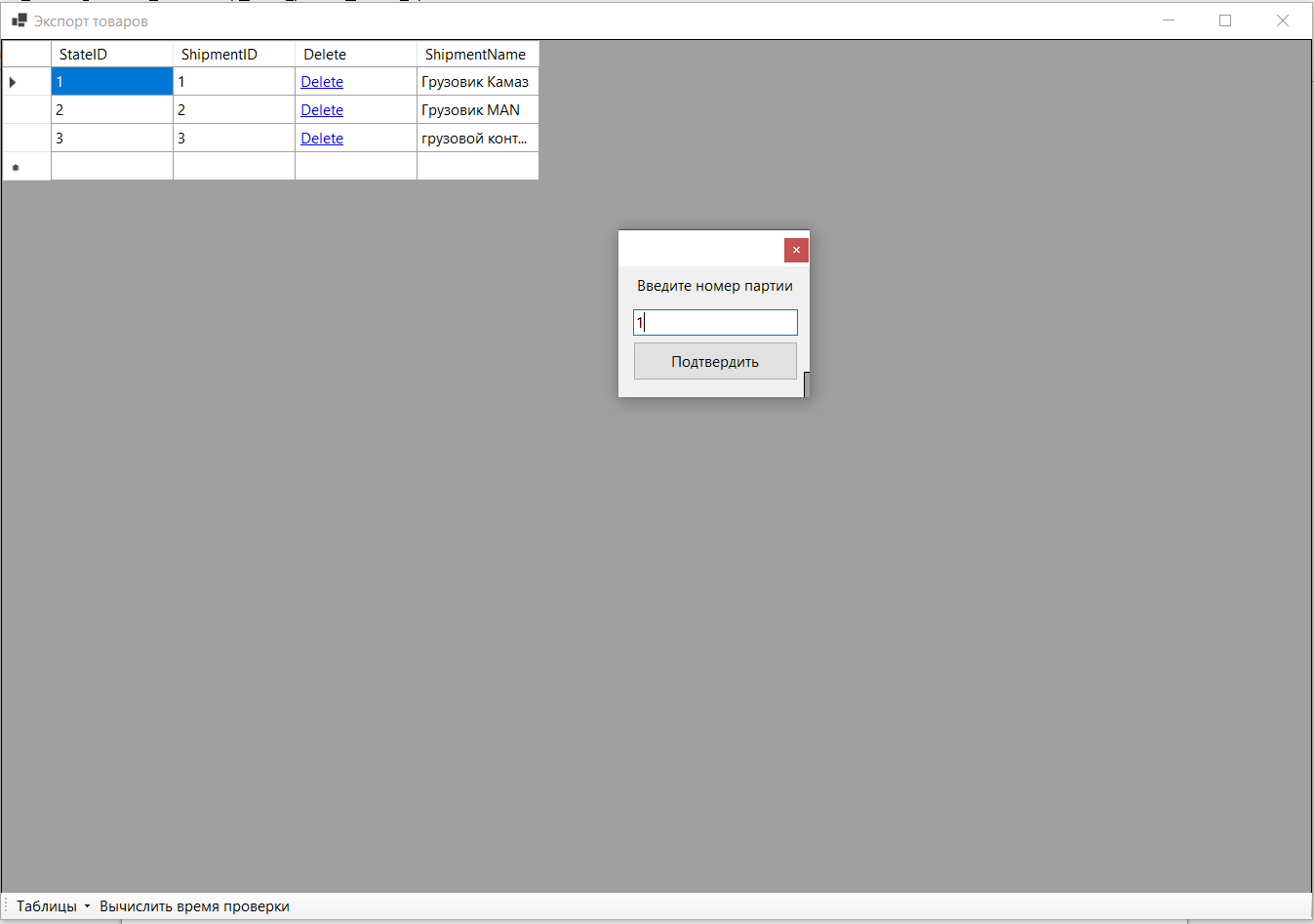


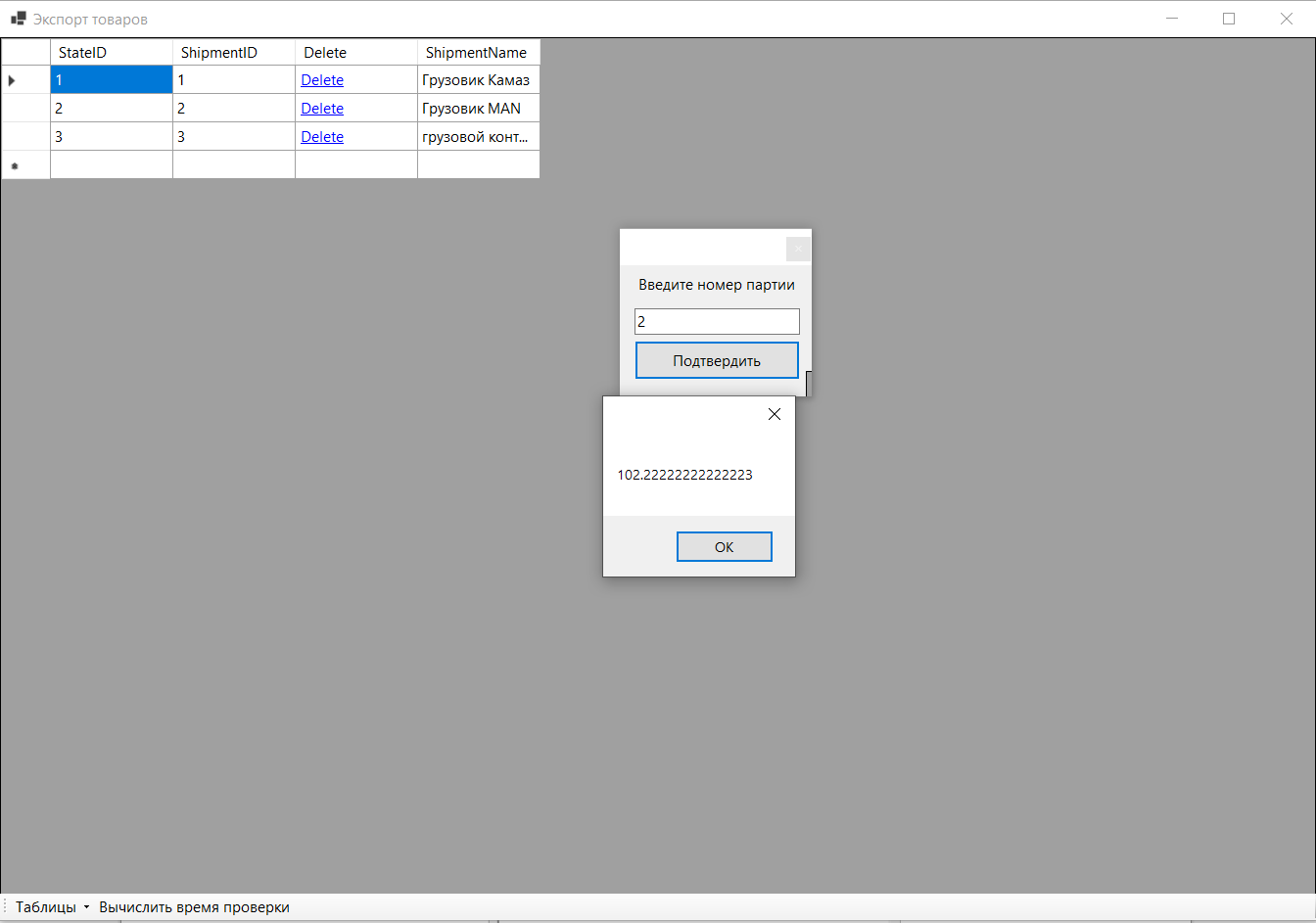
*Рисунок 5 – интерфейс программы.*

Приложение способно вычислять количество часов, требуемое таможне для проверки всех оставшихся товаров в поставке, для вызова этой функции достаточно нажать на кнопку «Вычислить время проверки», в таком случае программа предложит ввести номер поставки и затем выведет примерное количество часов до конца проверки поставки. Использование этой функции изображено на рисунках 6 и 7.

**Вывод**

В результате разработки была создана реляционная база данных для системы «Экспорт товаров» и приложение к ней.

*Рисунок 6 — окно выбора номера партии для вычислений*

*Рисунок 7 — окно вывода количества часов*

Тестирование программы:

Программа успешно выводит таблицы базы данных, а также изменяет их по желанию пользователя. Придусмотрена защита от ввода неподходящих под поля символов. Также программа вычисляет часы работы таможни без сбоев.

**Вывод**

Разработанная система базы данных и приложения безошибочно в рамках выявленных требований и ограничений обрабатывает запросы пользователя.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы по разработке реляционной базы данных экспорта товаров с использованием СУБД MySQL и разработке приложения на Windows Forms с использованием языка программирования C# были достигнуты значительные результаты и получен ценный опыт.

Была спроектирована и создана эффективная реляционная база данных, отражающая полную структуру таможенной системы с использованием СУБД MySQL. В базе данных предусмотрены все необходимые таблицы, отношения, индексы и ключи для эффективного хранения и обработки информации.

Было разработано приложение на Windows Forms с использованием языка программирования C#, которое взаимодействует с созданной базой данных. Приложение предоставляет интерфейс для работы с данными экспорта товаров, позволяет добавлять, редактировать и удалять информацию о поставках, товарах и прочих сущностях.

Разработанное приложение соответсвтует своим требованиям. Приложение успешно справилось со всеми поставленными задачами и продемонстрировало высокую производительность при работе с базой данных.

Полученный опыт в проектировании баз данных, разработке приложений на Windows Forms и взаимодействии с СУБД MySQL является ценным и позволит применить навыки и знания в будущей профессиональной деятельности.

СПИСОК ЛИТЕРАТУРЫ

1. Джуба С., Волков А. Д40 Изучаем PostgreSQL 10 / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2019. – 400 с.: ил. ISBN 978-5-97060-643-8
2. Дейт, К. Дж. Д27 Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме", 2005. — 1328 с.: ил. — Парал. тит. англ. ISBN 5-8459-0788-8 (рус.)
3. Бьюли А. Изучаем SQL. – Пер. с англ. – СПб: Символ­Плюс, 2007. – 312 с., ил. ISBN­13: 978­5­93286­051­9 ISBN­10: 5­93286­051­0

ПРИЛОЖЕНИЕ

**Программный код для создания приложения для базы данных**

using System.Diagnostics;

using System.Data;

using System.Data.SqlClient;

using MySql.Data.MySqlClient;

namespace Customs

{

public partial class Form1 : Form

{

private MySqlConnection sqlConn;

private MySqlCommandBuilder sqlBuilder;

private MySqlDataAdapter sqlData;

private DataSet dataSet;

private string table = "State";

private int tableRows = 2;

private bool newAdding = false;

public Form1()

{

InitializeComponent();

}

private void LoadData()

{

try

{

sqlData = new MySqlDataAdapter($"SELECT \*, 'Delete' AS 'Delete' FROM {table}", sqlConn);

sqlBuilder = new MySqlCommandBuilder(sqlData);

sqlBuilder.GetInsertCommand();

sqlBuilder.GetUpdateCommand();

sqlBuilder.GetDeleteCommand();

dataSet = new DataSet();

sqlData.Fill(dataSet, table);

dataGridViewMain.DataSource = dataSet.Tables[table];

for (int i = 0; i < dataGridViewMain.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridViewMain[tableRows, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ОШИБКА СТОП", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadData()

{

try

{

if (dataSet == null)

throw new Exception("dataset == null!");

dataSet.Tables[table].Clear();

sqlData.Fill(dataSet, table);

dataGridViewMain.DataSource = dataSet.Tables[table];

for (int i = 0; i < dataGridViewMain.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridViewMain[tableRows, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ОШИБКА СТОП", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Form1\_Load(object sender, EventArgs e)

{

string arg = "server=localhost;user=root;database=Kurs;";

sqlConn = new MySqlConnection(arg);

try

{

sqlConn.Open();

LoadData();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ОШИБКА СТОП", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void StripButtonReload\_Click(object sender, EventArgs e)

{

ReloadData();

}

private void dataGridViewMain\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == tableRows)

{

string task = dataGridViewMain.Rows[e.RowIndex].Cells[tableRows].Value.ToString();

if (task == "Delete")

{

int rowIndex = e.RowIndex;

dataGridViewMain.Rows.RemoveAt(rowIndex);

dataSet.Tables[table].Rows[rowIndex].Delete();

sqlData.Update(dataSet, table);

}

else if (task == "Insert")

{

int rowIndex = dataGridViewMain.Rows.Count - 2;

DataRow row = dataSet.Tables[table].NewRow();

if (table == "State")

{

row["StateID"] = dataGridViewMain.Rows[rowIndex].Cells["StateID"].Value;

row["StateName"] = dataGridViewMain.Rows[rowIndex].Cells["StateName"].Value;

}

else if (table == "BatchType")

{

row["BatchTypeID"] = dataGridViewMain.Rows[rowIndex].Cells["BatchTypeID"].Value;

row["TypeName"] = dataGridViewMain.Rows[rowIndex].Cells["TypeName"].Value;

}

else if (table == "Batch")

{

row["BatchID"] = dataGridViewMain.Rows[rowIndex].Cells["BatchID"].Value;

row["BatchTypeID"] = dataGridViewMain.Rows[rowIndex].Cells["BatchTypeID"].Value;

row["ShipmentID"] = dataGridViewMain.Rows[rowIndex].Cells["ShipmentID"].Value;

row["Volume"] = dataGridViewMain.Rows[rowIndex].Cells["Volume"].Value;

}

else if (table == "Customs")

{

row["CustomsID"] = dataGridViewMain.Rows[rowIndex].Cells["CustomsID"].Value;

row["StateID"] = dataGridViewMain.Rows[rowIndex].Cells["StateID"].Value;

row["CustomsLocation"] = dataGridViewMain.Rows[rowIndex].Cells["CustomsLocation"].Value;

row["CustomsName"] = dataGridViewMain.Rows[rowIndex].Cells["CustomsName"].Value;

}

else if (table == "Process")

{

row["ProcessID"] = dataGridViewMain.Rows[rowIndex].Cells["ProcessID"].Value;

row["CustomsID"] = dataGridViewMain.Rows[rowIndex].Cells["CustomsID"].Value;

row["BatchTypeID"] = dataGridViewMain.Rows[rowIndex].Cells["BatchTypeID"].Value;

row["Volume"] = dataGridViewMain.Rows[rowIndex].Cells["Volume"].Value;

row["Finished"] = dataGridViewMain.Rows[rowIndex].Cells["Finished"].Value;

}

else if (table == "Schedule")

{

row["ScheduleID"] = dataGridViewMain.Rows[rowIndex].Cells["ScheduleID"].Value;

row["CustomsID"] = dataGridViewMain.Rows[rowIndex].Cells["CustomsID"].Value;

row["ShipmentID"] = dataGridViewMain.Rows[rowIndex].Cells["ShipmentID"].Value;

row["Time"] = dataGridViewMain.Rows[rowIndex].Cells["Time"].Value;

row["Date"] = dataGridViewMain.Rows[rowIndex].Cells["Date"].Value;

}

else if (table == "Shipment")

{

row["ShipmentID"] = dataGridViewMain.Rows[rowIndex].Cells["ShipmentID"].Value;

row["StateID"] = dataGridViewMain.Rows[rowIndex].Cells["StateID"].Value;

row["ShipmentName"] = dataGridViewMain.Rows[rowIndex].Cells["ShipmentName"].Value;

}

dataSet.Tables[table].Rows.Add(row);

dataSet.Tables[table].Rows.RemoveAt(dataSet.Tables[table].Rows.Count - 1);

dataGridViewMain.Rows.RemoveAt(dataGridViewMain.Rows.Count - 2);

dataGridViewMain.Rows[e.RowIndex].Cells[tableRows].Value = "Delete";

sqlData.Update(dataSet, table);

newAdding = false;

}

else if (task == "Update")

{

int r = e.RowIndex;

if (table == "State")

{

dataSet.Tables[table].Rows[r]["StateID"] = dataGridViewMain.Rows[r].Cells["StateID"].Value;

dataSet.Tables[table].Rows[r]["StateName"] = dataGridViewMain.Rows[r].Cells["StateName"].Value;

}

else if (table == "BatchType")

{

dataSet.Tables[table].Rows[r]["BatchTypeID"] = dataGridViewMain.Rows[r].Cells["BatchTypeID"].Value;

dataSet.Tables[table].Rows[r]["TypeName"] = dataGridViewMain.Rows[r].Cells["TypeName"].Value;

}

else if (table == "Batch")

{

dataSet.Tables[table].Rows[r]["BatchID"] = dataGridViewMain.Rows[r].Cells["BatchID"].Value;

dataSet.Tables[table].Rows[r]["BatchTypeID"] = dataGridViewMain.Rows[r].Cells["BatchTypeID"].Value;

dataSet.Tables[table].Rows[r]["ShipmentID"] = dataGridViewMain.Rows[r].Cells["ShipmentID"].Value;

dataSet.Tables[table].Rows[r]["Volume"] = dataGridViewMain.Rows[r].Cells["Volume"].Value;

}

else if (table == "Customs")

{

dataSet.Tables[table].Rows[r]["CustomsID"] = dataGridViewMain.Rows[r].Cells["CustomsID"].Value;

dataSet.Tables[table].Rows[r]["StateID"] = dataGridViewMain.Rows[r].Cells["StateID"].Value;

dataSet.Tables[table].Rows[r]["CustomsLocation"] = dataGridViewMain.Rows[r].Cells["CustomsLocation"].Value;

dataSet.Tables[table].Rows[r]["CustomsName"] = dataGridViewMain.Rows[r].Cells["CustomsName"].Value;

}

else if (table == "Process")

{

dataSet.Tables[table].Rows[r]["ProcessID"] = dataGridViewMain.Rows[r].Cells["ProcessID"].Value;

dataSet.Tables[table].Rows[r]["CustomsID"] = dataGridViewMain.Rows[r].Cells["CustomsID"].Value;

dataSet.Tables[table].Rows[r]["BatchTypeID"] = dataGridViewMain.Rows[r].Cells["BatchTypeID"].Value;

dataSet.Tables[table].Rows[r]["Volume"] = dataGridViewMain.Rows[r].Cells["Volume"].Value;

dataSet.Tables[table].Rows[r]["Finished"] = dataGridViewMain.Rows[r].Cells["Finished"].Value;

}

else if (table == "Schedule")

{

dataSet.Tables[table].Rows[r]["ScheduleID"] = dataGridViewMain.Rows[r].Cells["ScheduleID"].Value;

dataSet.Tables[table].Rows[r]["CustomsID"] = dataGridViewMain.Rows[r].Cells["CustomsID"].Value;

dataSet.Tables[table].Rows[r]["ShipmentID"] = dataGridViewMain.Rows[r].Cells["ShipmentID"].Value;

dataSet.Tables[table].Rows[r]["Time"] = dataGridViewMain.Rows[r].Cells["Time"].Value;

dataSet.Tables[table].Rows[r]["Date"] = dataGridViewMain.Rows[r].Cells["Date"].Value;

}

else if (table == "Shipment")

{

dataSet.Tables[table].Rows[r]["ShipmentID"] = dataGridViewMain.Rows[r].Cells["ShipmentID"].Value;

dataSet.Tables[table].Rows[r]["StateId"] = dataGridViewMain.Rows[r].Cells["StateID"].Value;

dataSet.Tables[table].Rows[r]["ShipmentName"] = dataGridViewMain.Rows[r].Cells["ShipmentName"].Value;

}

sqlData.Update(dataSet, table);

dataGridViewMain.Rows[e.RowIndex].Cells[tableRows].Value = "Delete";

}

}

ReloadData();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ОШИБКА СТОП", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridViewMain\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newAdding == false)

{

newAdding = true;

int lastRow = dataGridViewMain.Rows.Count - 2;

DataGridViewRow row = dataGridViewMain.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridViewMain[tableRows, lastRow] = linkCell;

row.Cells["Delete"].Value = "Insert";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ОШИБКА СТОП", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridViewMain\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newAdding == false)

{

int rowIndex = dataGridViewMain.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dataGridViewMain.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridViewMain[tableRows, rowIndex] = linkCell;

editingRow.Cells["Delete"].Value = "Update";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ОШИБКА СТОП", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void stateToolStripMenuItem\_Click(object sender, EventArgs e)

{

table = "State";

tableRows = 2;

newAdding = false;

LoadData();

}

private void batchToolStripMenuItem\_Click(object sender, EventArgs e)

{

table = "Batch";

tableRows = 4;

newAdding = false;

LoadData();

}

private void batchTypeToolStripMenuItem\_Click(object sender, EventArgs e)

{

table = "BatchType";

tableRows = 2;

newAdding = false;

LoadData();

}

private void customsToolStripMenuItem\_Click(object sender, EventArgs e)

{

table = "Customs";

tableRows = 4;

newAdding = false;

LoadData();

}

private void processToolStripMenuItem\_Click(object sender, EventArgs e)

{

table = "Process";

tableRows = 4;

newAdding = false;

LoadData();

}

private void scheduleToolStripMenuItem\_Click(object sender, EventArgs e)

{

table = "Schedule";

tableRows = 5;

newAdding = false;

LoadData();

}

private void shipmentToolStripMenuItem\_Click(object sender, EventArgs e)

{

table = "Shipment";

tableRows = 3;

newAdding = false;

LoadData();

}

private void toolStripDropDownButton1\_Click(object sender, EventArgs e)

{

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

new Form2().Show();

}

private void dataGridViewMain\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

e.Control.KeyPress -= new KeyPressEventHandler(Column\_KeyPress);

if (dataGridViewMain.CurrentCell.ColumnIndex == 0)

{

TextBox textBox = e.Control as TextBox;

if(textBox != null)

{

textBox.KeyPress += new KeyPressEventHandler(Column\_KeyPress);

}

}

}

private void Column\_KeyPress(object sender, KeyPressEventArgs e)

{

if(!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

}

}

using MySql.Data.MySqlClient;

using MySqlX.XDevAPI.Relational;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Customs

{

public partial class Form2 : Form

{

private MySqlConnection sqlConn;

private MySqlDataAdapter sqlData;

private DataSet dataSet;

private DataSet dataSetSec;

public Form2()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

int j = 0;

bool flag = false;

int speed = 0;

double final = 0;

int ShipNum = Convert.ToInt32(textBox1.Text);

sqlData = new MySqlDataAdapter($"SELECT \* FROM Batch WHERE ShipmentID = {ShipNum}", sqlConn);

dataSet = new DataSet();

sqlData.Fill(dataSet, "Batch");

sqlData = new MySqlDataAdapter("SELECT \* FROM Process", sqlConn);

dataSetSec = new DataSet();

sqlData.Fill(dataSetSec, "Process");

for (int i = 0; i < dataSet.Tables["Batch"].Rows.Count; i++)

{

while (!flag)

{

if(Convert.ToInt32(dataSetSec.Tables["Process"].Rows[j]["BatchTypeId"]) == Convert.ToInt32(dataSet.Tables["Batch"].Rows[i]["BatchTypeID"]))

{

speed = Convert.ToInt32(dataSetSec.Tables["Process"].Rows[j]["Volume"]);

flag = true;

}

j++;

}

final += (double)Convert.ToInt32(dataSet.Tables["Batch"].Rows[i]["Volume"])/(double)speed;

j = 0;

flag = false;

}

MessageBox.Show(final.ToString());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ОШИБКА СТОП", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Form2\_Load(object sender, EventArgs e)

{

string arg = "server=localhost;user=root;database=Kurs;";

sqlConn = new MySqlConnection(arg);

try

{

sqlConn.Open();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ОШИБКА СТОП", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}